

# Introduction to Object Oriented Programming – CS12320

## First Contact

Chris Loftus

[cwl@aber.ac.uk](mailto:cwl@aber.ac.uk)

E38 (top of Comp Sci building)

1

## Welcome!

- First session is a quick overview of what's to follow...
- But first, by May (the end of this course) you should be able:
  - to write simple object-oriented programs using the Java language;
  - to design those programs using standard notations;
  - demonstrate professionalism by writing high-quality code

2

## What can we all expect?

- There are no lectures as such
- Everything, except tutorials happens in the lab
- YOU have to be here!
- I, plus demonstrators will be here!
- If you want extra help/lectures let me know
- There is a tutorial every week, starting week after next
- Sometimes we will use Qwizdom .....

3



What do you think is the single most important reason that we're using Quizdom? [qvr.qwizdom.com](http://qvr.qwizdom.com) Q5VN94

1. *Promotes your ability to communicate your ideas.*
2. *Strengthens your ability to debate and defend your answers, and to suggest improvements.*
3. *Promotes peer discussion that is balanced, with ideas put forth evenly from both partners.*
4. *Promotes a safe environment for you to answer what you honestly think, rather than answering what you think the instructor wants.*
5. *Gives you feedback on how well you understand a topic.*
6. *Gives me as your instructor feedback on what needs to be taught better, or expanded upon.*
7. *Encourages you to mentally engage with the concepts so that the lecture is not just passive listening and note taking.*

4

## Assessment

- No exam!
- Mini-assignments (40%):
  - Mini-assignment 1 (20%): Set of small worksheets running throughout the semester...
  - Mini-assignment 2 (20%): 26 February – 11 March
- Individual main assignment (50%):
  - 7 March – 6 May
- 9 tutorials (10%)

5

## Recommended book

- See Aspire Reading List on Blackboard, but:
  - Sierra, K., Head First Java, O'Reilly, 2005
  - In library, old but...
  - Feel free to use a different book

6

Which of the following have you done before? (Choose all that apply)



- A) Created a web page
- B) Written programs in a programming language
- C) Written your own operating system
- D) None of the above

## The aims of this session

- Write and run your first Java programs!
- Use an integrated development environment (IDE) called BlueJ to help you do this that illustrates features of Object-Orientation...
  - I will talk a bit and then you have a go...
  - Demonstrators are here to help...
  - Work in pairs to help each other...

## Where are the course materials?

On Blackboard (let's take a look):

- You will find sections, including:
- Schedule with links to:
  - Panopto recordings
  - Lead-in course (THIS)
  - Workshop talks (slides)
  - Worksheets
  - Tutorials
- The assignments under the Assignments link
- Discussion forums
- Twitter stream: #java\_cs12320 to ask questions etc

9

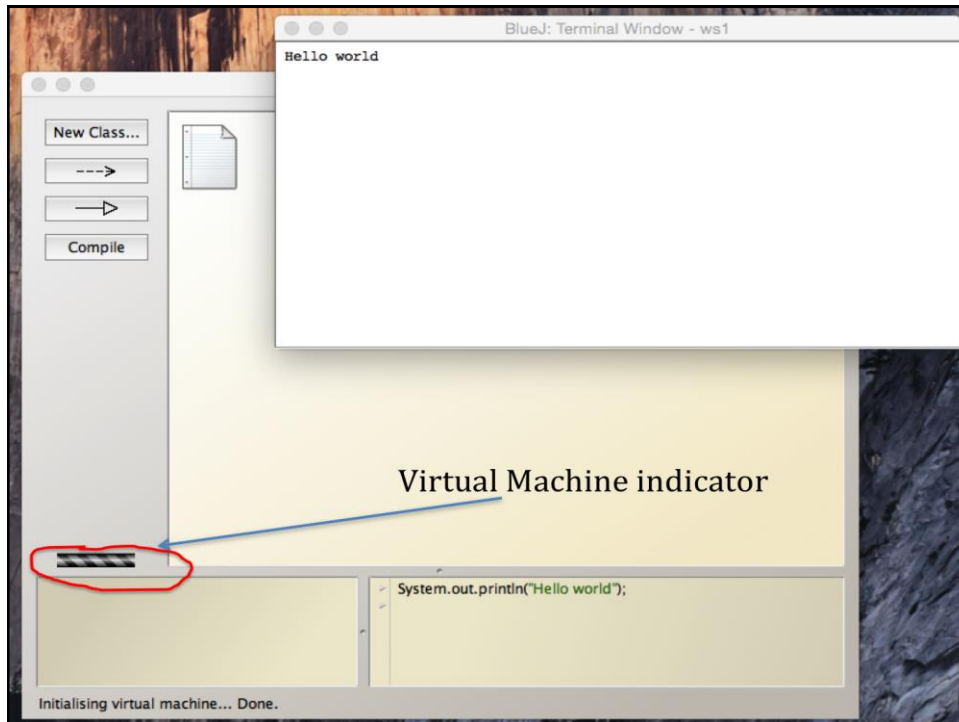
## The Hello World program

- In a moment the worksheet will get you to do the same...
- Demo of me:
  - Starting BlueJ (may need to go to View to show codepad)
  - Entering and running:

```
System.out.println("Hello world");
```

within the BlueJ codepad

10



## Your turn:

- If not done so, download CS12320-leadin.zip from BB and unzip on M: drive
- Try to get *hello world* using *BlueJ* to work (follow **instructions** on worksheet 1).
- Then try other variations listed
  - Use print rather than println
  - Add “\n” characters within the text to be displayed
  - Add spaces in the text to be displayed
  - What happens if you leave out the “;”?

How far have you got with  
worksheet 1 etc?



- A) Not started
- B) Got “hello world” to work
- C) Used print rather than println
- D) Tried taking out “\n”
- E) Added spaces and tried out missing ;

Creating objects and running  
methods

## The fundamental idea of object-oriented design and programming

- You break the problem up using the THINGS in it. These are called **objects**, and are examples of **classes**.
- You then manipulate those objects.
- This is different from procedural design and programming where you break the problem up more by what HAPPENS.

15

## Variables

```
int age = 18;  
int newAge = 0;  
newAge = age;
```

18  
int age

18  
int newAge

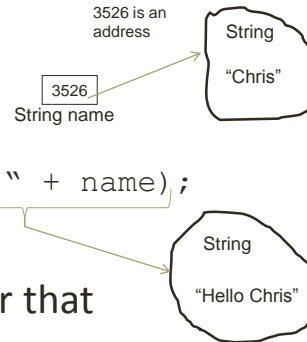
- Revision: Q What is a variable?
- In Java there are two main kinds: those that hold primitive values (integers, real numbers, booleans) and those that hold references to objects (String, Person etc)... I'll say more later

16



- So an example would be:

```
String name = "Chris";
System.out.println("Hello " + name);
```



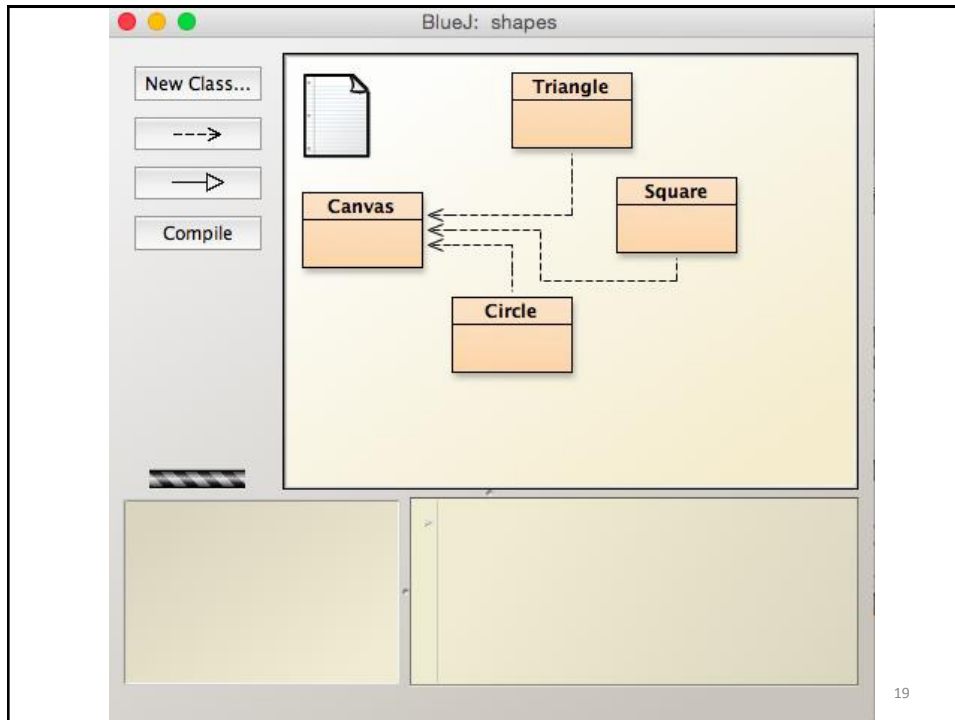
- Will display Hello Chris
- Note the special "+" operator that concatenates two strings...
- `name` will be an object reference variable, Since it's a String, to something that holds characters.
- Let me demo this in BlueJ...

17

## Objects and classes: The Shapes program using BlueJ

- Let's use BlueJ in the way it was intended...
- We are now going to import some ready made classes that allow us to create shapes.
- This demo and the following worksheet looks at:
  - Opening the existing Shapes project
  - A brief introduction to the class concept
  - How does a class relate to an object?
  - Using variables that have references to shape objects

18



## Classes and objects (briefly)

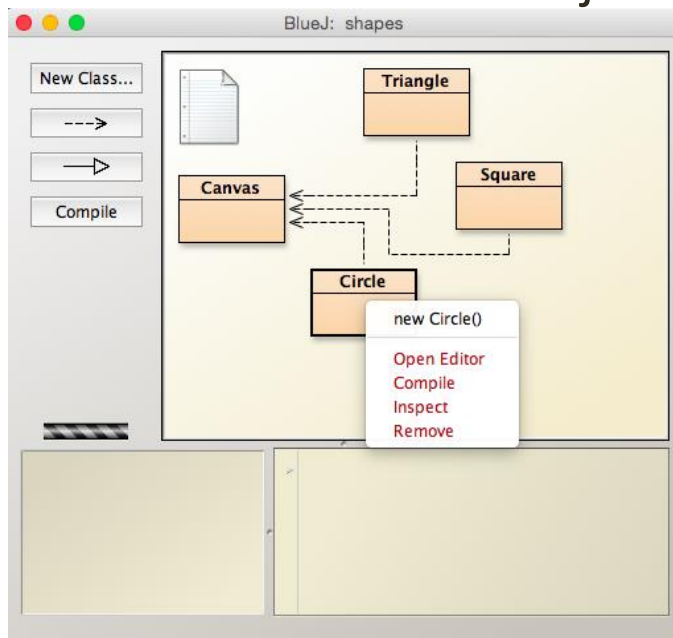
- A class is like a template (or jelly mould, or blueprint) from which you make lots of objects that all have the same kinds of values but those values can be different...
- We're all familiar with classes (kinds) of things: animals, cars....
- So here there's a Circle class from which you can create lots of circle objects, each with a different color...
- Let's create some Circle objects...

## Classes vs objects analogy!

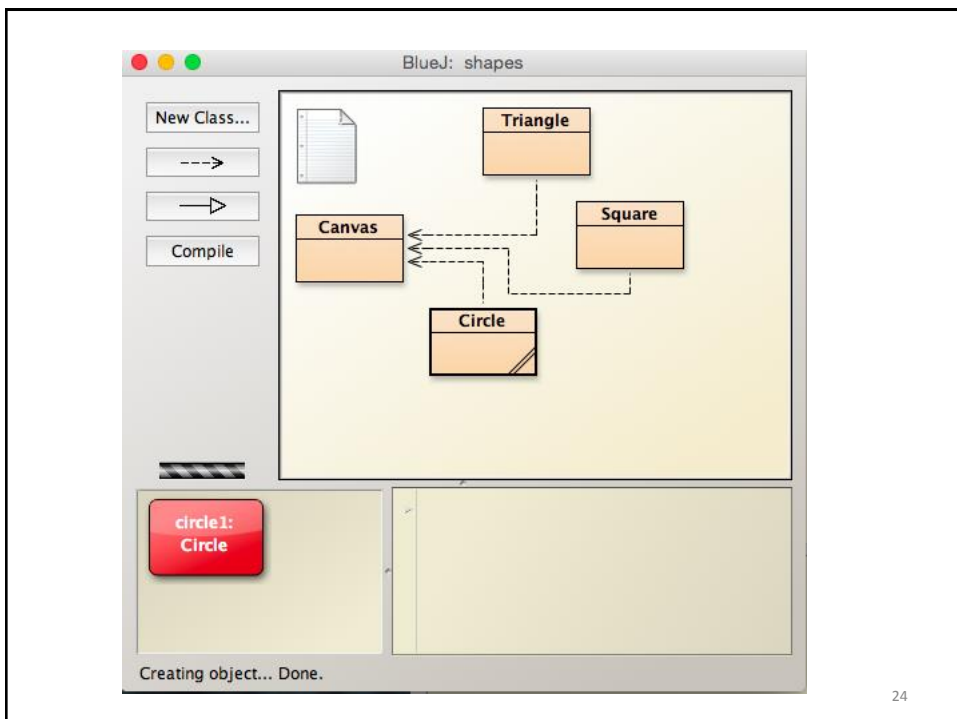
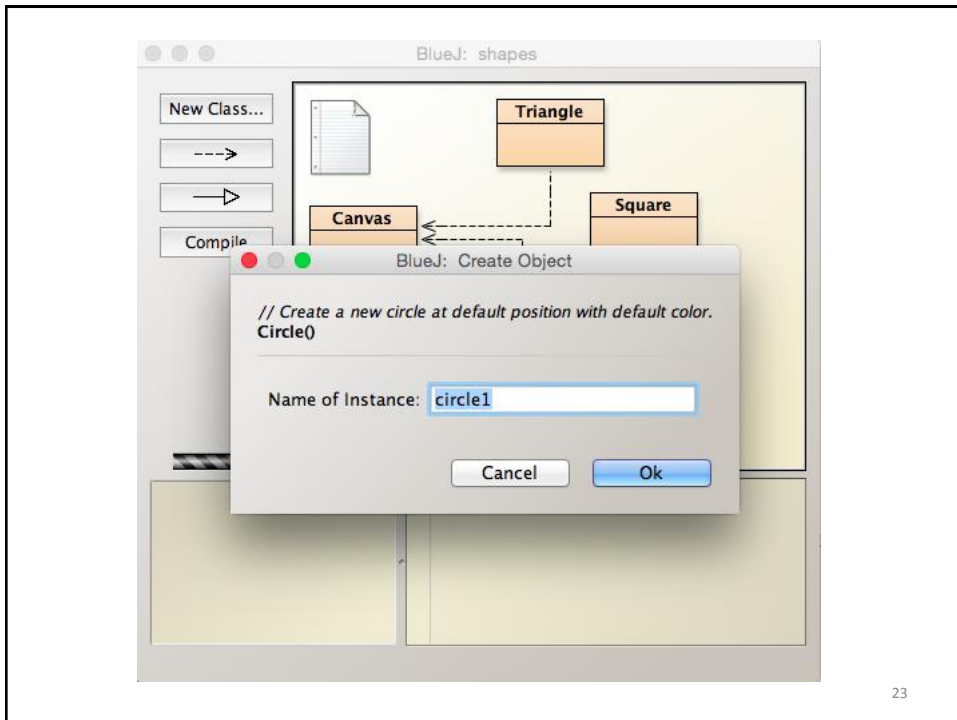


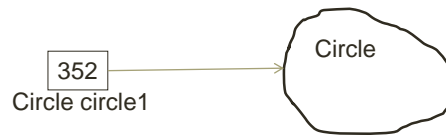
21

## Let's create a Circle object



22

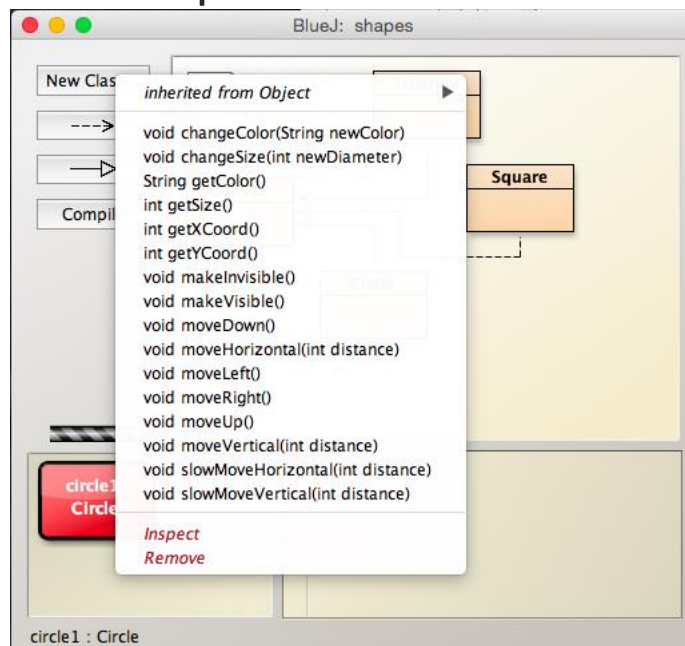




- `circle1` is the name of a variable that holds a reference to some place in memory that contains the object...
- Let's create a second one and give in the variable `circle2`...
- What's interesting is that we can do operations on circles. If I right-click the circle...

25

## The operations on circles



26

## In Java operations/functions are called **methods**

- They are defined as part of the class...
- They are applied to objects, e.g.  
`circle1.makeVisible();`
- Note the syntax:  
–`variable.method(para1,paraN);`
- Parameters are optional and carry data into the object...
- Let's see how this works to create two circles with different colors...

27

We can also use codepad to perform methods...

The screenshot shows the BlueJ IDE interface. On the left, there's a sidebar with buttons: "New Class...", "---->", "---->", and "Compile". The main workspace displays a class hierarchy diagram with boxes for "Triangle", "Canvas", "Square", and "Circle". "Canvas" is the superclass for "Triangle", "Square", and "Circle". Below the diagram, there are two buttons labeled "circle1: Circle" and "circle2: Circle". To the right, a window titled "BlueJ Shapes Demo" shows two overlapping circles, one red and one blue. At the bottom, a code editor shows the following code:

```

circle1.makeVisible();
circle1.moveRight();
circle2.makeVisible();
circle2.changeColor("red");

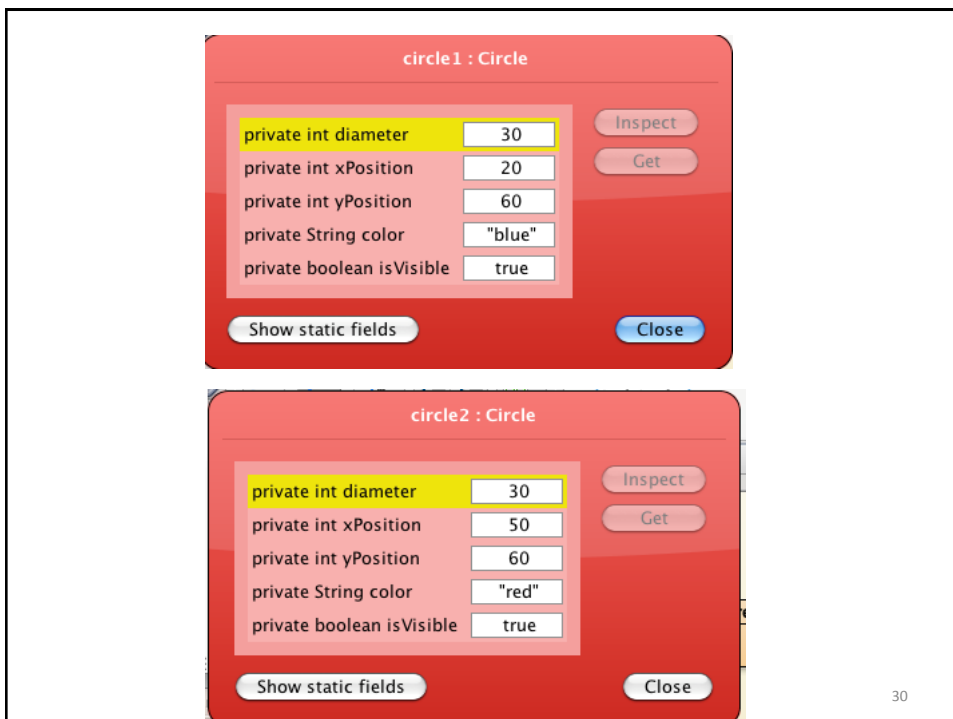
```

A red arrow points from the word "parameter" to the string "red" in the last line of code. At the bottom left, a status bar says "Creating object... Done." and the page number "28" is visible at the bottom right.

## Objects have properties called instance variables/fields in Java

- We call the values associated with objects instance variables of the object. We can inspect these instance variables with `inspect` selected from a right click on the object
- Let's see how...

29



30

## Your turn:

- Try to get BlueJ to draw a cartoon character out of shapes (follow instructions on worksheet 2).
- Would be a good idea to store your code in a .txt file as before so you can modify it and rerun it and also you'll need it later

31

How far have you got with  
worksheet 2?



- A) Not started
- B) Completed compiling step 1
- C) Completed codepad steps 1-2
- D) Completed codepad step 3 - drawn
- E) Completed codepad step 4 - changed
- F) Completed codepad step 5 - cartoon



Quiz: What is a variable?  
(Choose one answer)



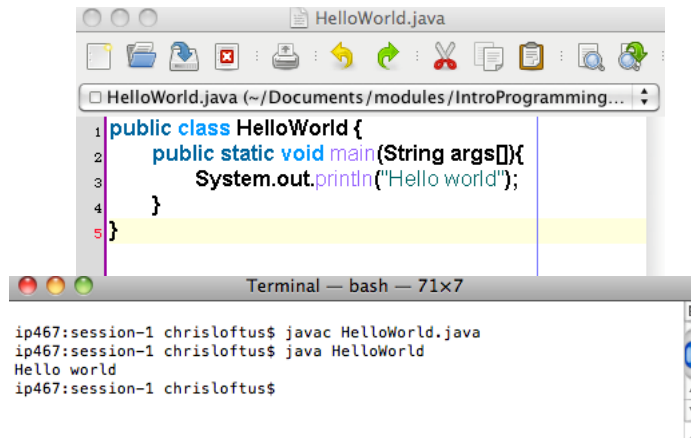
- A) An area of memory that can hold a value that never changes
- B) An object instance of a class
- C) An area of memory that can be assigned a value
- D) I don't know

Quiz: In object-orientation, what is a method? (Choose the best answer)



- A) Code that is executed on an object
- B) It is the main way of defining a class
- C) Code that is executed on an object, but that never changes the state of that object
- D) I don't know

Sneak Peek: there *are* main programs that we can use to manipulate objects, but for now we are manipulating objects without



The screenshot shows a code editor window titled 'HelloWorld.java' with the following code:

```
1 public class HelloWorld {  
2     public static void main(String args[]){  
3         System.out.println("Hello world");  
4     }  
5 }
```

Below the code editor is a terminal window titled 'Terminal — bash — 71x7' showing the following commands and output:

```
ip467:session-1 chrisloftus$ javac HelloWorld.java  
ip467:session-1 chrisloftus$ java HelloWorld  
Hello world  
ip467:session-1 chrisloftus$
```

35

Ifs

36

## Conditional tests

- We need more kinds of statement in a programming language...
 

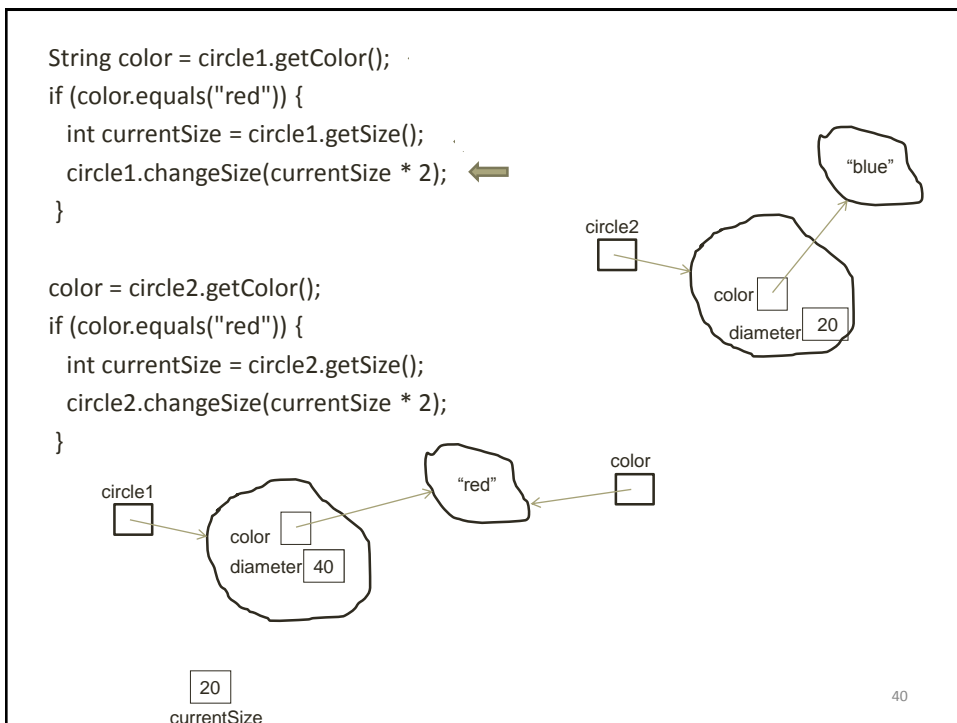
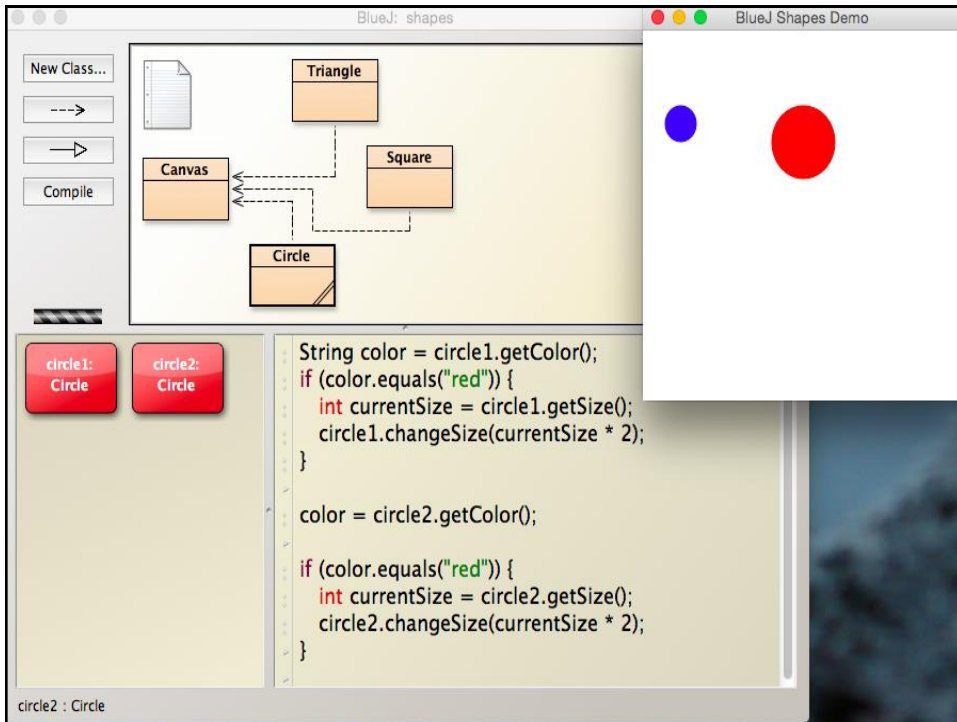
```
statement1;
statement2;
...
```
- Let's say we had a strange game where we always made sure that red circles were twice their default creation size. Other colors remain unchanged...
- We want to say, **if the shape is circle and its color is red** then find its current diameter, multiply by 2 and change the diameter to that new size...

37

## Conditional tests

- This is a conditional statement that enables you to do various things depending on a value being **true** or **false**
- Let's try a simple example
  - Here I simply get the color of circle1 and then circle2
  - If one of them is red then I double its size
  - Really we should read data from the keyboard, but we'll do that later

38



## if statement

```
if (condition is true) {
    do stuff;
}
```

Boolean expression that evaluates to true or false

Only done if conditional was true

```
else if(condition is true){
    do alternative stuff;
}
```

The "else" parts are optional: either further tests or a final else catch all

```
else {
    do stuff if nothing else was done!;
}
```

41

## What kind of tests can you do?

- Whether two items are the same:

```
if (color.equals("red"))...
if (currentSize == 30)...
```

- Whether an item is greater (or less) than another:

```
if (currentSize > 30)           Also >=
if (currentSize < 30)           Also <=
```

- Whether different:

```
if (!color.equals("red"))...    !true == false
if (currentSize != 30)...       !false == true
```

- Whether two things are true:

```
if (color.equals("red") && currentSize > 30)...
```

- Whether at least one thing is true:

```
if (color.equals("red") || currentSize > 30)...
```

42

## Your turn:

- Try to get BlueJ to change the size of the circle using **if** statements to decide what to do based on color (**follow instructions on worksheet 3**).

43

How far have you got with  
worksheet 3?



- A) Not started
- B) Completed steps 1-3
- C) Completed step 4
- D) Completed step 5
- E) Completed step 6

## What is printed? (Choose one)



```
int yourAge = 18;  
int chrisAge = 48;
```

```
if (yourAge > chrisAge) {  
    System.out.println("You are REALLY old!");  
}  
else if (yourAge < chrisAge) {  
    int diff = chrisAge - yourAge;  
    System.out.println("You are " + diff + " years younger than Chris");  
}  
else {  
    System.out.println("You're just old");  
}
```

- A) You are 30 years younger than Chris
- B) You are REALLY old!
- C) You're just old
- D) I don't know

## What is printed? (Choose one)



```
int yourAge = 49;  
int chrisAge = 48;
```

```
if (yourAge > chrisAge) {  
    System.out.println("You are REALLY old!");  
}  
else if (yourAge < chrisAge) {  
    int diff = chrisAge - yourAge;  
    System.out.println("You are " + diff + " years younger than Chris");  
}  
else {  
    System.out.println("You're just old");  
}
```

- A) You are 30 years younger than Chris
- B) You are REALLY old!
- C) You're just old
- D) I don't know

## What is printed? (Choose one)



```
int yourAge = 17;
boolean gotPermission = false;

if (yourAge >= 18) {
    System.out.println("You're old enough to decide for yourself");
}
else if ((yourAge >= 16 && yourAge < 18) && gotPermission) {
    System.out.println("You can stay out until 10pm");
}
else {
    System.out.println("You must be home by 6pm!");
}
```

- A) You must be home by 6pm!
- B) You're old enough to decide for yourself
- C) You can stay out until 10pm
- D) I don't know

## What is printed? (Choose one)



```
int yourAge = 15;
boolean gotPermission = true;

if (yourAge >= 18) {
    System.out.println("You're old enough to decide for yourself");
}
else if ((yourAge >= 16 && yourAge < 18) || gotPermission) {
    System.out.println("You can stay out until 10pm");
}
else {
    System.out.println("You must be home by 6pm!");
}
```

- A) You must be home by 6pm!
- B) You're old enough to decide for yourself
- C) You can stay out until 10pm
- D) I don't know



# Loops

49

# Loops

- As well as testing for a condition, we want to be able to loop repeating something...
- One way to do this in Java is:  

```
while (condition) { do some stuff }
```
- This is very similar to an **if** statement, except it is done *repeatedly* until the condition is NOT **true** any more (this means that the condition needs to be changed within the "do some stuff" part, or you will never exit the loop).

50

## Demo of while loop

- Problem: I want to be able to randomly move a circle around the canvas 20 times
- Let's run my implementation...
- Solution (in pseudo-code):

```

get the width and height of the canvas;
count = 0; currentXPos = 0; currentYPos = 0;
while (count < 10){
    xPos = get random number in range 0..width-1;
    yPos = get random number in range 0..height-1;
    circle1.slowMoveHorizontal(xPos - currentXPos);
    circle1.slowMoveVertical(yPos - currentYPos);
    currentXPos = get circle1 X position;
    currentYPos = get circle1 Y position;
    count = count + 1;
}

```

51

## Simple example of while loop (Java has other kinds of loop too)

The screenshot shows a Java Swing application window titled "circle1 : Circle". On the left, there is a control panel with buttons for "Compile", "Run Tests", "recording", "End", and "Cancel". The main area of the window is divided into two parts. The top part is a diagram showing three classes: "Canvas", "Square", and "Circle". "Canvas" is connected to "Square" and "Circle" by dashed lines. The bottom part is a code editor showing the following Java code:

```

Canvas c = Canvas.getCanvas();
int width = c.getWidth();
int height = c.getHeight();
int xPos = 0;
int currentXPos = 0;
int yPos = 0;
int currentYPos = 0;
java.util.Random rand = new java.util.Random();
circle1.makeVisible();
int count = 0;
while (count < 10){
    xPos = rand.nextInt(width);
    yPos = rand.nextInt(height);
    currentXPos = circle1.getXCoord();
    currentYPos = circle1.getYCoord();

    circle1.slowMoveHorizontal(xPos - currentXPos);
    circle1.slowMoveVertical(yPos - currentYPos);
    count = count + 1;
}

```

The status bar at the bottom of the window shows "ircle1 : Circle" and the page number "52".

## What just happened?

- The Canvas class has a method called `getCanvas()`. This is another way of getting the Canvas object...
  - Q. What's the other way you've seen up until now?
- I need the canvas to find its width and height...
- Lots of declarations of variables...

53

- Use of random number generator
  - `rand.nextInt(width)` gives a pseudo-random number in range `0 .. width-1`
- The while loop condition is checked when Java tries to enter the loop. If `true`, then the body of the loop is run, else if `false` Java jumps to the next statement after the end of the loop...
- Q. What if I forgot to increment `count`?

54

## Algorithm and pseudo-code

- As your programs get larger, it gets harder to just write down the code lines that accomplish what you want
- Instead, you need to break the program down into bigger steps and then figure out how to do each step
- If it is still too hard to program, you can repeat this process
- If you write down the separate steps, then you have an algorithm for your program
- Typically, we express this using structured natural language called pseudo-code

55

```
while (more employees) {  
    get employee details;  
    calculate pay;  
    pay employee;  
}
```

56

## Your turn

- In this exercise repeat what I just did (follow worksheet 4, step 1 - 5) and some more (step 6).

57

How far have you got with  
worksheet 4?



- A) Not started
- B) Completed steps 1-4
- C) Completed step 5
- D) Completed step 6

## Getting ready for CS12320

- BlueJ gives you a nice 'feeling' for OO programming, but it is not a production type of tool
- From next week we will be using the Eclipse IDE
- It is much more powerful to use than BlueJ when we come to creating classes and running and debugging them via a main program

59

## Where is the main program?

- Before we talk about creating your own classes – let's look at this

60

## Setting up a main program

- A main program creates some objects and then makes them do something
- It also has to live inside a class though
- The structure looks like:

```
public class Test {
    public static void main(String args[]) {
        Circle c=new Circle();
        c.makeVisible();
        c.slowMoveHorizontal(100);
    }
}
```

61

## Create a Test class (all together)

- Use BlueJ to create a Test class (as part of the Shape project)
- Delete all its code and replace with Test.txt
- Compile
- Run this by right clicking the main method of the CLASS Test (you can also run from command line: `java Test`)
- Any code that we put in the code pad could be put inside a main program. Try

```
Circle c=new Circle(); //the general way of creating an object
c.makeVisible();
c.slowMoveHorizontal(100);
```

62

Once we are all done, we'll go on

63

Defining your own class

64



Programming = **data+algorithm** (Knuth)

- Java is **object-oriented** – so the **data** is the main way things are organised – e.g. a game is an **object**
- Then you get the objects to do things – e.g. `game.play()`
- That in turn involves getting other objects to do things – e.g. `player1.move()`
- The data consists of objects which are designed by using **classes**

## Notice how this is different from CS12020

- There the problem was mainly broken up ('decomposed') by what happened
- Here the first decomposition is by data
- Then those data things do stuff

## Classes and Objects (again)

- Objects are the things in a program
- Classes are PATTERNS for designing these things (or Blueprints, or Templates, or ...)
- All objects in a class will **have** some attributes (**instance variables**) These are 'private'
- And be able to **do** some things (**methods**) These are 'public' but how they work is private

## Classes and objects (briefly)

- A class is like a template (or jelly mould, or blueprint) from which you make lots of objects that all have the same kinds of values but those values can be different...
- We're all familiar with classes (kinds) of things: animals, cars....
- So we've been using a Circle class from which you can create lots of circle objects, each with a different color...

## Classes vs objects analogy!

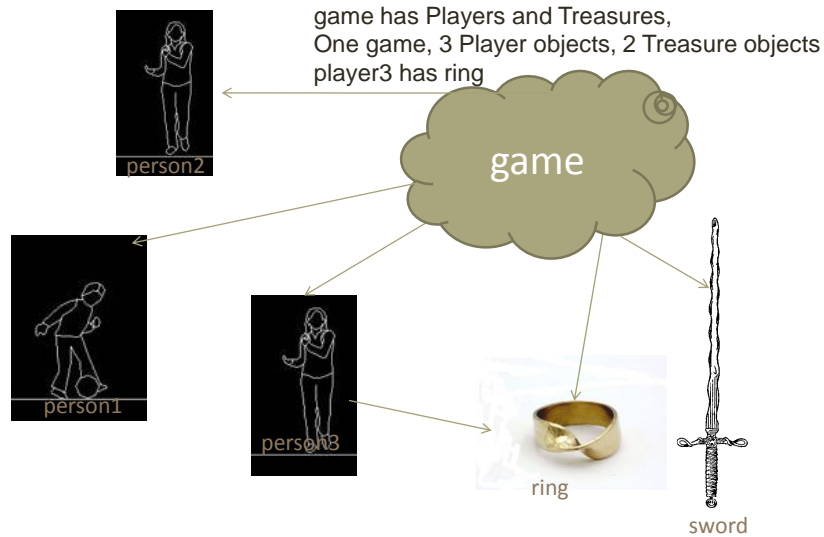


69

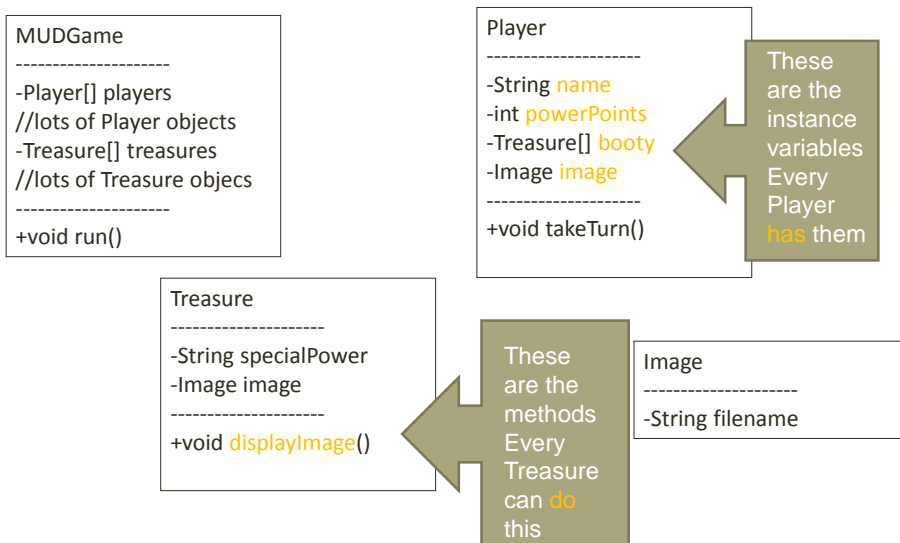
## So, suppose I want to write a Multi-Player role playing game

- What are the **kinds** of things that are likely to show up when you play the game??
- These may well be your classes
- The individual things will be your objects

## Object Diagram (as program runs)



## Class diagram (for design ignore details)



## Think about a Cartoon class

- Think about what each Cartoon object can **do**
- And about what each Cartoon object would **have**

73

## So, what does a Cartoon class look like?

- You will be able to **do** things with Cartoon objects  
`Cartoon c = new Cartoon(); //General Java way`  
`c.draw();`
- A Cartoon object will **have** some shapes in it

74

## Look at worksheet 5

- I have given you something like a Cartoon, a class called Picture
- Follow the instructions for the first part of worksheet5 to **create the Picture class**
- Create an object of class Picture and draw it
- Inspect your object
- Change it to Black and White and reinspect
- Look at the code for Picture. Can you see the instance variables and methods?
- Then follow the instructions to test your Picture in a **main program**

75

## If time, create your own class

- Create a Cartoon class based on Picture
- Modify your Test class to draw a Cartoon
- It would be nice to be able to draw the Cartoon anywhere
- From CS120 and what we have done you should have the skills to do that

76

How far have you got with  
worksheet 5?



- A) Not started
- B) Completed first section 1-10
- C) Completed second section 11-17
- D) Completed third section 18-20
- E) Completed step 21

## What's next?

- This was a brief overview of some features in Java and also some programming concepts in general...
- In the remainder of the course we will explore many of these things and more in much more detail...